Dando um passeio no ASP.NET

Por Mauro Sant'Anna (mas_mauro@hotmail.com). Mauro é um "MSDN Regional Director", consultor e instrutor.

O principal objetivo da arquitetura .NET é permitir ao usuário o fácil acesso a seus aplicativos e dados em qualquer lugar, a qualquer hora e usando qualquer dispositivo. Para concretizar esta visão existem vários componentes, de servidores a ferramentas de desenvolvimento.

Uma importante capacidade desta arquitetura é a de interagir com navegadores HTML comuns, até mesmo produzidos por outras empresas que não a Microsoft. É aqui que o ASP.NET entra em cena. O ASP.NET é uma ferramenta "RAD" para o desenvolvimento de aplicativos baseados em páginas HTML. Dentre suas várias vantagens, é um grande salto em produtividade quando comparado a qualquer outra ferramenta do mercado. Ele torna o desenvolvimento de aplicativos baseados em Web tão simples como o desenvolvimento de aplicativos em Visual Basic. O programador não tem sequer que conhecer HTML ou "scripts de cliente" (usualmente JScritp) para criar aplicativos sofisticados.

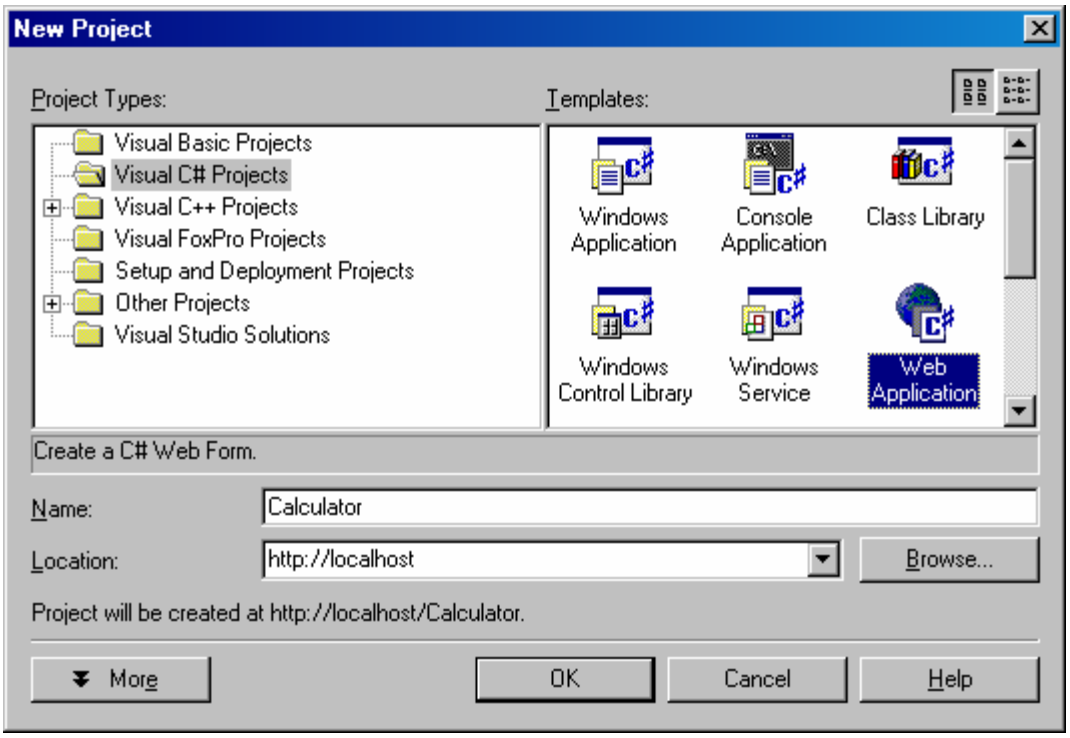
A chave para este recurso é um ambiente de desenvolvimento que usa componentes para processar eventos do navegador e gerar HTML. Além dos componentes que já vem com o VS.NET, certamente aparecerão diversos componentes de terceiros.

Mesmo que você não decida abraçar toda a arquitetura .NET (WinForms, por exemplo), os ganhos de produtividade do ASP.NET são razão bastante para usa-lo.

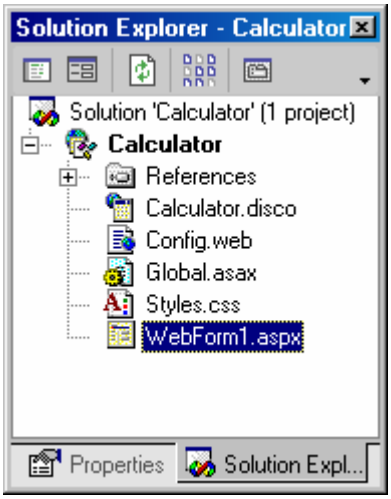
De forma a mostrar como é fácil escrever um aplicativo HTML que inclui até mesmo scripts de cliente, vou guia-lo no desenvolvimento de um aplicativo ASP.NET bastante simples: uma calculadora de quatro operações. Vou partir do princípio que você já tem o "Visual Studio Beta 1" instalado e funcionando. Usaremos a linguagem C#, mas esta demonstração é essencialmente a mesma para todas as linguagens da arquitetura.NET.

Criando um projeto

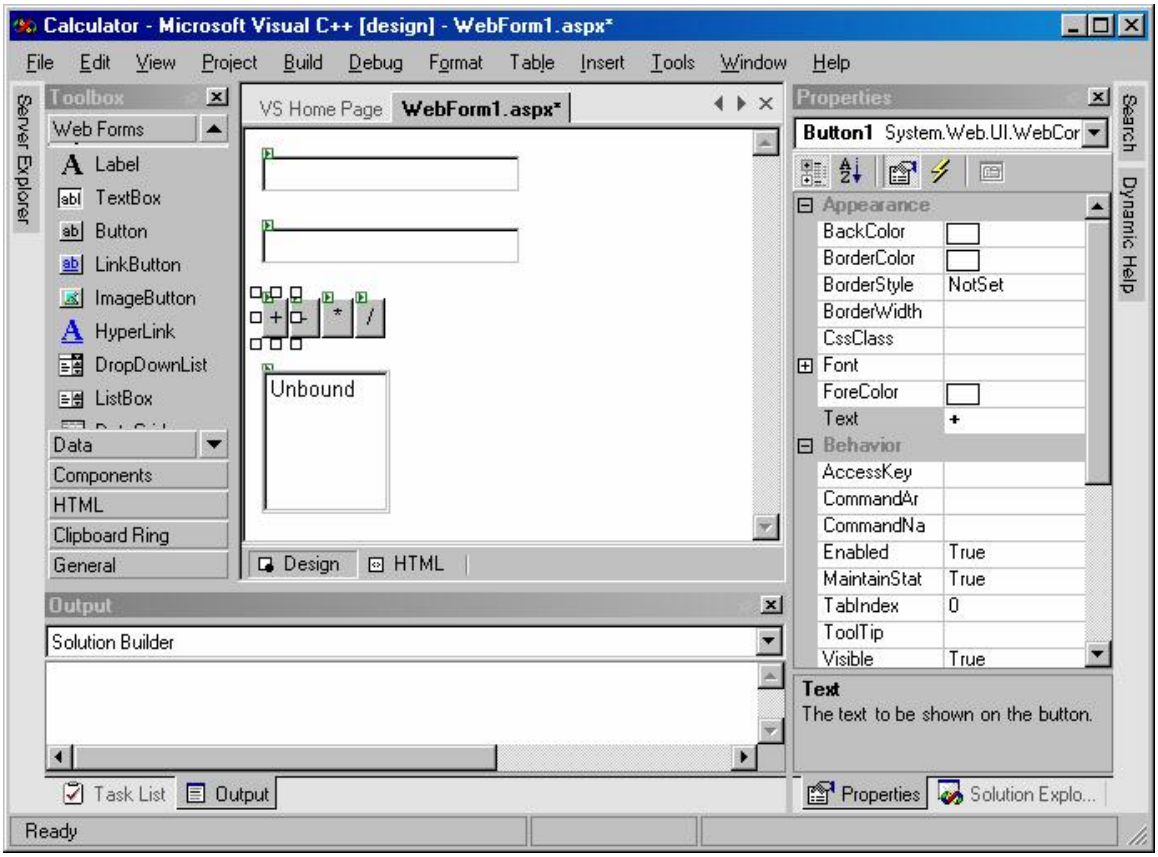
Abra o VS.NET e selecione "New | Project | Visual C# Projects | WebApplication". Dê nome "Calculator" e aperte "Ok":



Você agora tem uma "solução" (um conjunto de projetos relacionados) com um único projeto dentro. O projeto contém um único "WebForm" chamado "WebForm1.aspx", uma página HTML especial que tem um fonte em linguagem de alto nível associado – C# no caso. Não se preocupe com os demais arquivos:



Abra agora o "Toolbox" (Ctrl-Alt-X), vá até a página "WebForm1.aspx" e arraste dois componentes "TextBox", quatro "Button" e um "ListBox". Use a tecla "Enter" para separar os componentes em diferentes linhas. Selecione a página "Properties" (F4) e mude a propriedade "Text" dos botões para as quatro operações ("+", "-", "*" e "/"), como mostrado a seguir:

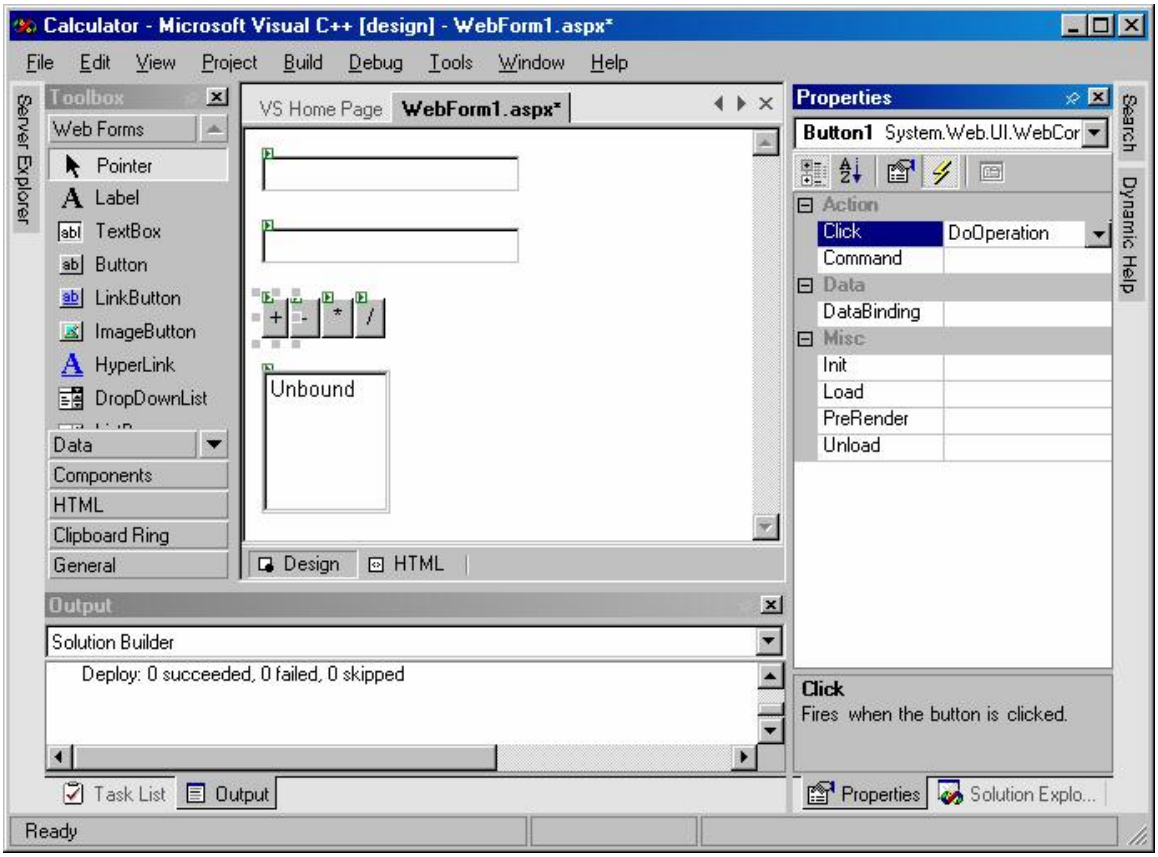


Note que você pode escolher entre dois modos de “Page Layout”: “linear” e “grid”. O modo “GridLayout” permite o posicionamento dos componentes em coordenadas absolutas “X e Y”, mas não é compatível com todos os navegadores. Deixaremos esta propriedade com o valor padrão, “LinearLayout”.

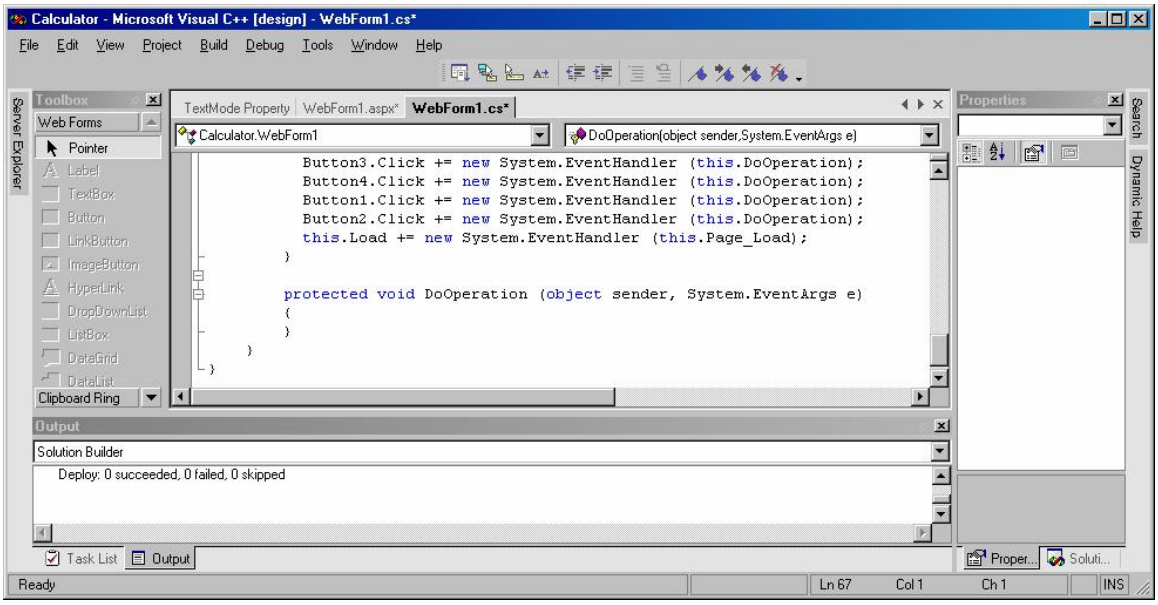
Os componentes que colocamos no formulário se parecem muito com “tags” HTML, mas isto não é verdade. Por exemplo, o componente “TextBox” pode gerar as tags “INPUT”, “INPUT PASSWORD” ou “TEXTAREA”, dependendo do valor da propriedade “TextMode”. Eles podem até mesmo não gerar nenhuma saída visível no navegador caso a propriedade “Visible” seja “false”. Os componentes são abstrações de alto nível de elementos usados nos aplicativos. Os próprios componentes geram a saída em HTML conforme necessário e de forma compatível com o navegador (HTML 4.0, HTML 3.2, WAAP, JScript etc). O seu aplicativo não precisa saber nada a respeito de HTML; os componentes interagem com o navegador específico que o usuário estiver rodando.

Selecione agora o botão “+” e clique no “raio” na página “Properties”. Note que cada componente tem vários eventos. Esta é uma maneira de trabalho muito parecida com a do Visual Basic, mas é um grande contraste em relação a outras ferramentas HTML, onde todo o formulário gera um único evento e você deve escrever código para descobrir que componente fez o que.

Escreva “DoOperation” na caixa de texto ao lado do “Click”:



Quando você pressionar “Enter”, será exibido o editor de código. Como desejamos que todos os botões apontem para o mesmo evento, vá até o WebForm (clique na aba “WebForm1.aspx” na parte superior) e selecione todos os botões pressionando a tecla “Shift”. Vá até a página de eventos, clique no “combobox” ao lado de “Click” e selecione “DoOperation”. Novamente o editor de código será aberto, conforme mostrado a seguir:

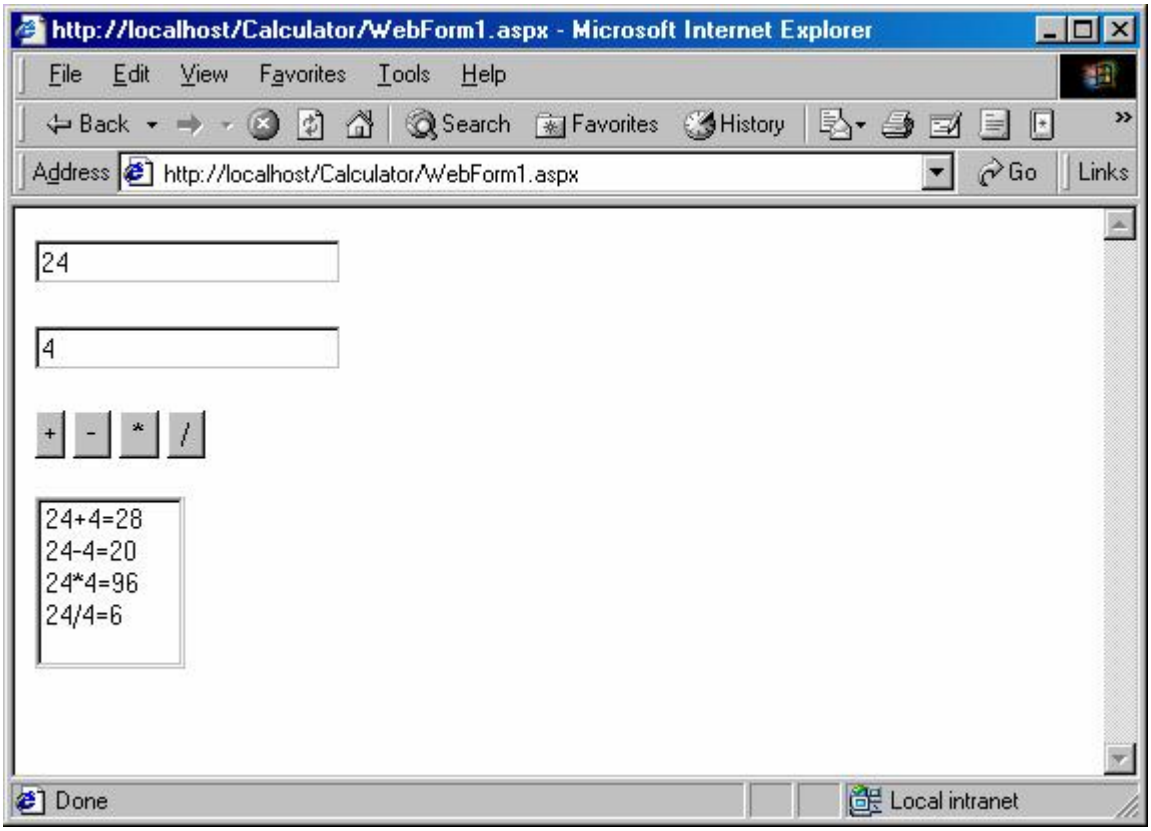


Existe um detalhe interessante a respeito do gerenciamento de eventos: o nome do método associado ao evento é completamente arbitrário; ele não é baseado em nenhuma "regra de nomes". Dentre as várias vantagens, isto permite que mapeemos vários eventos ao mesmo método, como acabamos de fazer.

Digite agora o seguinte código:

```
protected void DoOperation (object sender, System.EventArgs e) {
  // Verifica se a validação foi correta
  if (!this.IsValid)
    return;
  // Pega o texto do botão
  string Oper = ((Button)sender).Text;
  // Variável para o resultado. O tipo decimal é menos propenso a erros de arredondamento
  decimal Result = 0;
  string Msg = "";
  // Este é um bloco de tratamento de erro
  try {
  // Pega o conteúdo dos campos de texto e converte para decimal
    decimal N1 = TextBox1.Text.ToDecimal();
    decimal N2 = TextBox2.Text.ToDecimal();
    // Você pode usar string em um a switch.As opções são mutuamente exclusivas
    switch (Oper) {
      case "+": Result = N1 + N2; break;
      case "-": Result = N1 - N2; break;
      case "*": Result = N1 * N2; break;
      case "/": Result = N1 / N2; break;
    }
    Msg = TextBox1.Text + Oper + TextBox2.Text + "=" + Result.ToString();
  }
  // Trata divisão poe zero
  catch(System.DivideByZeroException) {
    Msg = "Você não pode dividir por zero ";
  }
  // Trata erros de conversão
  catch(System.FormatException) {
    Msg = "Você deve fornecer dois números";
  }
  // Adiciona ao list box. Isto é MUITO mais simples que o ASP tradicional
  ListBox1.Items.Add(Msg);
}
```

Agora selecione "Debug | Start Without Debugging". O VS.NET irá compilar o seu código, atualizar os arquivos no servidor Web e abrir um navegador para chamar a sua página. Você pode ter alguns erros de compilação; corrija-os até conseguir a página de teste:



Existem várias coisas interessantes neste projeto:

- O modelo de desenvolvimento é muito mais parecido com o Visual Basic do que com outras ferramentas HTML.
- Não vimos nenhuma linha de código HTML. Você até pode editar o HTML diretamente (selecione "HTML" na página "WebFrom1.aspx"), mas isto não é necessário.
- O código (arquivo ".cs") é separado da apresentação (arquivo ".aspx"). O "Web designer" pode mudar a aparência da aplicação sem afetar o código.
- O seu programa ter performance de código compilado e seu fonte está seguro; seu provedor só enxerga os fontes do seu código se você quiser.
- Os componentes mantêm automaticamente o estado sem uma única linha de código. Imagine quanto código seria necessário apenas para manter o estado do "ListBox" acima.
- Temos o poder de uma linguagem "de gente grande" como o C# de forma a permitir a escrita de código mais seguro e poderoso. Não existe mais a necessidade de escrever componentes COM apenas para complementar coisas difíceis de fazer em VBScript ou JScript.

- Podemos usar o poder do C# e VS.NET para melhor organizar o código e dividi-lo entre os diversos membros de uma equipe.

Validação no cliente

Vamos avançar um pouco mais no nosso exemplo. Suponha que desejemos validar o conteúdo dos "TextBoxes" no cliente. Algo como se os números entrados são válidos, sem precisar ir e voltar ao servidor. Usando a tecnologia corrente de navegadores, isto só é possível com "scrips de cliente", usualmente escrito em JScript. Scripts de cliente são particularmente difíceis de escrever e depurar, mas o ASP.NET pode gera-los automaticamente para nós!

Da página "Web Forms" no "Toolbox", adicione um "CompareValidator" ao lado de cada "TextBox". Mude as seguintes propriedades em cada um deles:

- ErrorMessage: "Este não é um número válido"
- ControlToValid: TextBox1 ou TextBox2
- Operator: DataTypeCheck
- Type: Double

Rode o programa. Observe que os conteúdos dos TextBox serão validados no cliente, usando um "script" que foi enviado ao navegador. Você pode agora perguntar o que aconteceria caso tivesse sido usado um navegador que não suporte scripts de cliente. A resposta é simples: nenhum script é enviado e a validação será efetuada apenas no servidor. Na verdade, a validação sempre é feita no servidor de forma a evitar "falsificações de página" (spoofing). Esta é a razão para as duas linhas no início do nosso código: temos que impedir o resto de rodar caso a validação no servidor falhe.

Conclusão

O ASP.NET é um ambiente poderoso e produtivo para o desenvolvimento de aplicativos baseados em HTML. Ele torna o desenvolvimento de aplicativos Web tão simples como o desenvolvimento de aplicativos Windows usando uma ferramenta RAD como o Visual Basic.